УДК 004.415.2

Трибюк В.О., здобувач
Фант М.О., к.філол.н., доцент
Державний університет «Житомирська політехніка»

РЕАЛІЗАЦІЯ ПОВЕДІНКОВОГО ШАБЛОНУ СТРАТЕГІЯ ДЛЯ ОБРОБКИ ВЗАЄМОДІЙ КОРИСТУВАЧІВ З ОБ'ЄКТАМИ ПЛАТФОРМИ

Важливим аспектом будь-якої вебплатформи, де користувачі взаємодіють один з одним, є підтримка різних видів взаємодії, наприклад, коментарі, реакції на пости та інше. Саме тому виникає потреба у створенні алгоритмів, які дозволяють ізолювати спільну логіку для цих взаємодій. Одним із вирішень цієї проблеми є патерн Стратегія – це поведінковий шаблон, що групує схожі алгоритми, розміщує кожен з них в окремий клас та робить їх взаємозамінними. Основна його перевага полягає у зменшенні кількості операторів if-else, що робить систему більш гнучкою, наприклад, при додаванні нового типу об'єкта взаємодії. [1]

Розглядаючи вебплатформу для створення косметичних колекцій та ведення блогу, виділено три основних об'єкта взаємодії: пости, продукти та косметичні колекції. Користувач може виконати дві взаємодії з ними, а саме коментування та реакції. Тому в базі даних для цих об'єктів є дві таблиці Comments та Reactions, де є стовпчик, що визначає, до якого об'єкта користувач здійснив певну взаємодію. Саме ця властивість визначає, яку стратегію слід застосувати для обробки взаємодії. Реалізація відбувалася таким чином (рис. 1): перший крок – це створення інтерфейсу стратегії, так як один із трьох об'єктів не має за собою закріпленого власника, тому було необхідно створити два інтерфейси, один із яких розширює можливості іншого. Батьківський інтерфейс містить властивість, що визначає тип об'єкта, та метод, який перевіряє, чи може користувач взаємодіяти з об'єктом. Другий інтерфейс розширює батьківський та додає метод для отримання інформації про власника об'єкта. Наступним кроком потрібно реалізувати два узагальнені абстрактні класи, що слугують базовими для всіх трьох стратегій. Батьківський клас реалізує методи базового інтерфейсу і додає власні розширення, а саме, додає поле узагальненого репозиторію для роботи з базою даних, а також реалізує базову логіку перевірки існування сутності в таблиці. Другий узагальнений абстрактний клас наслідує батьківський клас і реалізує другий інтерфейс, але має обмеження на тип параметру, а саме він повинен наслідуватися від класу, що визначає ID користувача, як зовнішній

ключ. Клас додає поле репозиторію користувача та реалізує метод інтерфейсу, що використовує цей репозиторій для отримання інформації про об'єкт. Метод репозиторію є узагальненим і має те саме обмеження, що й клас. Далі, реалізовано три класи стратегії, що визначають поле типу об'єкту, а також два класи, змінюють логіку методу, який перевіряє чи можна взаємодіяти з об'єктом.

Завершальним кроком у реалізації патерна є створення контексту, який зберігає колекцію стратегій та повертає потрібну за визначеним типом сутності. Після цього у контейнері залежностей реєструються всі реалізації базового інтерфейсу, а також сама фабрика стратегій, яка відповідає за отримання правильного алгоритму з переліку зареєстрованих.

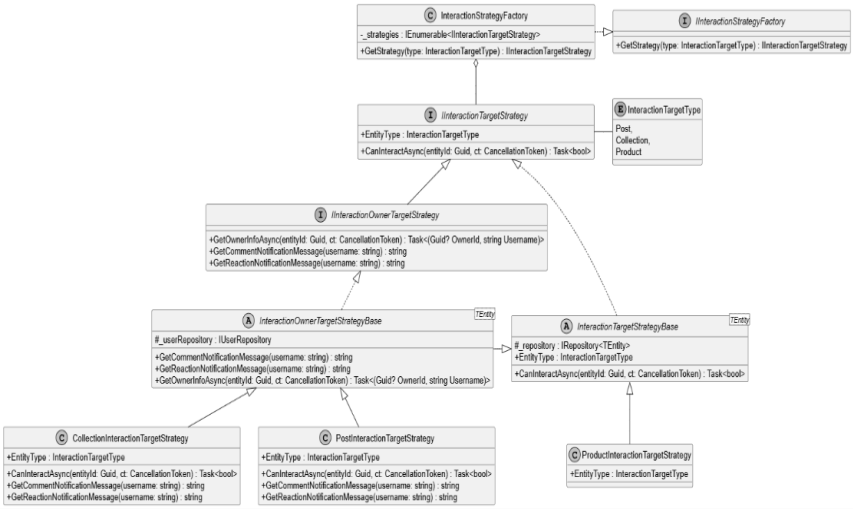

Рис. 1. Діаграма класів для реалізація патерна Стратегія для обробки користувацьких взаємодій

Отже обрана реалізація обробки взаємодій користувача з об'єктами платформи дозволяє ефективно керувати взаємодіями для різних типів об'єктів. Використання патерну Стратегія забезпечує чистіший та легкий в обслуговувані код.

Список використаних джерел:

1. Strategy design pattern. GeeksforGeeks. [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/system-design/strategy-pattern-set-1/.